A Bug's life

Finding and Managing Bugs
CSE 403
Lecture 23

What is a bug?

- Formally, a "software defect"
- SUT fails to perform to spec
- SUT causes something else to fail
- SUT functions, but does not satisfy usability criteria
- If the SUT works to spec and someone wants it changed, that's a feature request

What are the contents of a bug report?

- Repro steps – how did you cause the failure?
- Observed result – what did it do?
- Expected result – what should it have done?
- Any collateral information: return values/output, debugger, etc.
- Environment
  - Test platforms must be reproducible
  - "It doesn't do it on my machine"

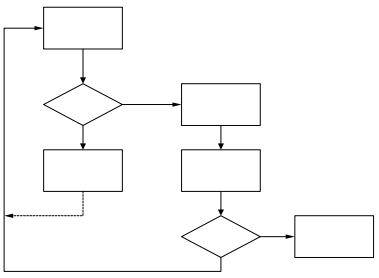
What makes a good bug report?

- Clear, descriptive title
- Accurate description of the problem
- Environment description
- Steps to reproduce the problem
  - Ideally, a minimal set of steps

Ranking bugs

- Severity
  - Sev 1: crash, hang, data loss
  - Sev 2: blocks feature, no workaround
  - Sev 3: blocks feature, workaround available
  - Sev 4: trivial (e.g. cosmetic)
- Priority
  - Pri 1: Fix immediately
  - Pri 2: Fix before next release outside team
  - Pri 3: Fix before ship
  - Pri 4: Fix if nothing better to do ☺

A Bug's Life



Bug Triage

- Decide which bugs to fix
- Reasons NOT to fix bugs
  - Ambiguous status of the bug
  - Cost of fix vs. benefit to the product
  - Risks that fixing it will cause other problems
    - Incorrect fix
    - Other portions of the code depend on incorrect behavior

Regression Testing

- Good: rerun the test that failed
  - Or write a test for what you missed
- Better: rerun related tests (e.g. component level)
- Best: rerun all product tests
  - Automation can make this feasible!

Tracking Bugs

- Raw bug count
  - Slope is useful predictor
- Ratio by ranking
  - How bad are the bugs we're finding?
- Find rate vs. fix rate
  - One step forward, two back?
- Management choices
  - Load balancing
  - Review of development quality

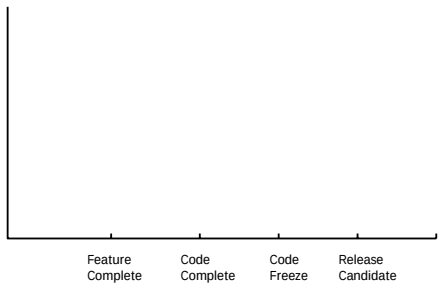
When can I ship?

- Test coverage sufficient
- Bug slope, find vs. fix lead to convergence
- Severity mix is primarily low-sev
- Priority mix is primarily low-pri

Milestones

- Feature complete
  - All features are present
- Code complete
  - Coding is done, except for the bugs
- Code Freeze
  - No more coding
- Release Candidate
  - I think it's ready to ship
- It's out the door

BUGs vs. Time



Basic entomology

- Some facts about bugs (from Code Complete)
  - The scope of most errors is limited
    - 85% could be fixed by modifying a single routine
  - Errors in assignment statements are common
    - 41% of errors in assignment statements
  - Most errors easy to fix
    - 85% a few hours
    - 14% a few hours to a few days
    - 1% multiple days
  - Old Microsoft Data (1992)
    - 10 to 20 defects per 1000 lines of code in test
    - 0.5 defects per 1000 lines of released code

Errors in testing

- Test cases can have errors too!
- Spending hours looking for bugs in code that turn out to be in the test case
- Test cases more prone to errors than the code
  - Especially when the developer writes the test case

When do you fix bugs?

- Write the code first, have someone else fix the bugs
- Fix blocking bugs, but save the minor stuff until code complete
- Aggressively fix bugs as they are found